AFRL-IF-RS-TR-2000-157
Final Technical Report
November 2000

TRANSMISSION CONTROL PROTOCOL (TCP) OVER ASYNCHRONOUS TRANSFER MODE (ATM): A SIMULATION STUDY

Bellcore

Teunis J. Ott, James E. Burns, and Larry H. Wong

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2000-157 has been reviewed and is approved for publication.

APPROVED: Priscilla A Cassidy

PRISCILLA A. CASSIDY
Project Engineer

FOR THE DIRECTOR:

WARREN H. DEBANY, Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** NOVEMBER 2000	**3. REPORT TYPE AND DATES COVERED** Final Sep 96 - Sep 98

4. TITLE AND SUBTITLE TRANSMISSION CONTROL PROTOCOL (TCP) OVER ASYNCHRONOUS TRANSFER MODE (ATM): A SIMULATION STUDY	**5. FUNDING NUMBERS** C - F30602-96-C-0260 PE - 62702F PR - 4519 TA - 22 WU - 44
6. AUTHOR(S) Teunis J. Ott, James E. burns, and Larry H. Wong	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Bellcore 445 South Street Morristown NJ 07960-6438	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/IFGA 525 Brooks Road Rome NY 13441-4514	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2000-157

11. SUPPLEMENTARY NOTES

Air Force Research Laboratory Project Engineer: Priscilla A. Cassidy/IFGA/(315) 330-1887

12a. DISTRIBUTION AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

The main body of this final report is a discussion of network studies performed using a simulation tool which was enhanced under this contract. The interaction between Transmission Control Protocol (TCP) and Asynchronous Transfer Mode (ATM), in particular Available Bit Rate (ABR), was studied.

14. SUBJECT TERMS Transmission Control Protocol ,TCP, Asynchronous Transfer Mode, ATM, Available Bit Rate, and ABR	**15. NUMBER OF PAGES** 60
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

Abstract

This is the final report for the "TCP-ATM Interactions" Project done at Bellcore in contract for Rome Laboratories, Sept. 1996-Sept. 1998. Among the deliverables were a simulation tool, a tutorial on ATM with ABR, a tutorial on TCP, and this final report.

This final report, among other things, reports on the results of network studies performed with the simulation tool.

1 Introduction

This is the final report of the project performed at Bellcore under contract F30602-96-C-0260 with the Department of the Air Force, through Rome Laboratories.

Section 2 of this report contains an abstract of the Statement of Work of this contract, with pointers to the corresponding sections in this report.

TCP (Transmission Control Protocol) is the protocol for reliable transport in the Internet. It relies on detection of packet loss for flow control. It also retransmits lost packets until a positive acknowledgement has been received. ABR (Available Bit Rate) is a service category in ATM (Asynchronous Transfer Mode) where VCs (Virtual Circuits) get explicit feedback on the rate (bandwidth) they can use. TCP is end-to-end and is window based, while ABR is restricted to the ATM part of the network and is rate based.

It is important to understand the interaction between TCP and ABR if TCP connections are (partially) transported over ABR VCs. For this purpose, a simulation tool was built at Bellcore. This simulation tool was rewritten and enhanced under contract with the Air Force. The tool is research quality software, meant for use by people with a background in Electrical Engineering or Computer Science. The simulation tool, with accompanying documentation, has been delivered to Rome Laboratories.

In addition, tutorials on ABR and on TCP were written and delivered to Rome laboratories.

The main body of this final report is a discussion of network studies performed with the simulation. The discussion in that part of this report assumes an understanding of TCP and ABR at the level of the tutorials mentioned above.

The tutorial on ABR (delivered in November 1997) also contains a description of a mechanism called "weighted Min-Max" that can be used (among other purposes) for giving priority to certain Virtual Circuits. This mechanism was invented at Bellcore for this project. A provisional patent application has been filed.

2 The Statement of Work

The first deliverable in this contract was a **simulation tool** with the following capabilities:

- **Priorities**

  The simulation tool can set weights for ABR VCs. The result of a higher weight is a (potentially) higher ACR (Allowed Cell Rate) and therefore higher throughput.

Results of performance studies with priorities are reported in Section 7.3 of this report.

- **Noisy Links**

  The simulation produced can make links (both "packet" or IP links and "cell" or ATM links) noisy. The drop probability can be made time-dependent, thus modeling links that change error characteristics over time ("correlated noise"). Results of performance studies are reported in Section 7.4 of this report.

- **Tutorials**

  A tutorial on ATM with ABR , see [2], was delivered to Rome Laboratories personnel prior to the meeting at Rome Laboratories in December 1997. A tutorial on TCP, see [3], was transmitted in July 1998. The tutorial on ATM with ABR contains both a tutorial and a description of the "Switch Behavior" for ABR developed at Bellcore for the simulation tool developed for the Air Force. A provisional patent application on the switch behavior developed for this purpose has been filed.

- **Network Designs**

  Simple networks were designed that make it easy to understand the impacts of features such as "Weighted ABR". Network design was done in cooperation with Rome Laboratories personnel.

- **Performance Studies**

  Additional performance studies described in Section 4.1.5 of the Statement of Work were done and are described in Sections 7.6 and 7.5 of this paper.

- **Software**

  A simulation tool for research on the interaction between TCP and ATM, in particular ABR, was developed. As agreed with Rome laboratories personnel, the software is research quality software, written in C++, with Perl scripts for model description.

Software and documentation were first handed over to Rome Laboratories Personnel in July 1997. Modified versions were delivered in December 1997 and January and July 1998. The latest version contains about 8,000 lines of C++ code and about 800 lines of Perl Script. As agreed with Rome laboratories personnel, the software is research quality software and does not follow the standards specified in MIL STD 498 as mentioned in requirement in Section 4.2.3 of the original Statement of Work.

In addition to the software package, a Software User Manual for the simulation package was written. The Software User Manual contains a complete description of how create a "Network File" (the file describing the network to be analyzed), how to run the simulation, how to set the "Mask" which controls what information is printed by the simulation, and to a certain degree how to interpret the output. Drafts of the software user manual were handed over to Rome Laboratories at the same time the code was handed over.

- **Licensing**

  The Statement of Work also describes the licensing arrangement for the software Bellcore delivers to Rome Labs. The licensing arrangement is not further discussed in this report.

- **Reporting**

  The Bellcore Technical leader of this project (Teunis J. Ott) and the Rome Laboratories representative (Priscilla Cassidy) had meetings in

  - October 1996 (Kickoff Meeting at Rome Air Force Base),
  - January 1997 (site visit to Bellcore),
  - February 1997 (at Bellcore, as side-trip to CECOM)
  - December 1997 (at Rome Laboratories)
  - June 1998 (at Bellcore)

A last meeting will be held in early September 1998.

In addition, Mr. Ott and Ms. Cassidy had frequent contacts by Email and by phone.

3 Motivation and Technical Background

A representative portion of a network that can be examined with the simulation tool is sketched in Figure 1. The simulation uses "packet connections" (TCP/IP connections between pairs of packet-hosts) and "cell-connections" (or "ATM-connections"), which we call VCs (Virtual Circuits). VCs are ABR VCs (Available Bit Rate VCs), UBR VCs (Unspecified Bit Rate VCs), VBR VCs (Variable Bit Rate) or CBR VCs (Constant Bit Rate VCs). CBR VCs are between a pair of Cell-Hosts, and go from one Cell-Host through the ATM Network to another Cell-Host. ABR VCs, UBR VCs and VBR VCs go from a segmentation/reassembly point in one edge-router through the ATM network to another segmentation/reassembly point in another edge-router. For ABR VCs we call those segmentation points "Virtual Sources" and the reassembly points "Virtual Destinations" (VSs and VDs). This use of the names VS and VD is consistent with, but not identical to, the use in the ATM Forum.

In a simulation, a packet-source (host) sends IP (data) packets to an edge-router where segmentation occurs. The cells of the packet are then handed over to a VC which carries them through the ATM Network to another edge-router, where reassembly occurs. The resulting IP packets are then sent to a packet-destination (host). The packet destination sends back acknowledgements (ACKs) in the form of smaller packets, which follow the same route back and also undergo segmentation and reassembly.

The (packet) source and destination use the TCP/IP Reno protocol to decide when to send, to acknowledge, or to resend a packet. The simulation contains a large number of possible "source models" for the behavior of the TCP sources. Of these, we use only two

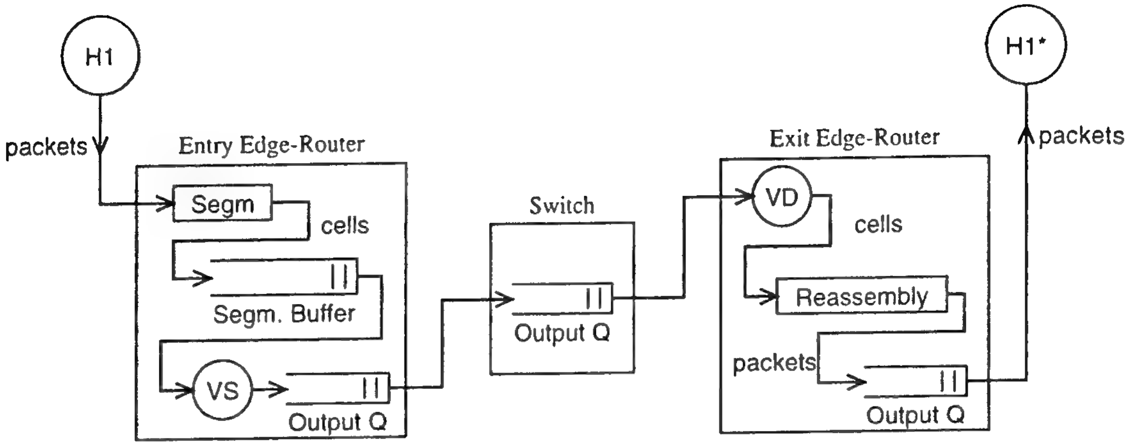

Figure 1: A Single TCP Connection

in the studies used in this report: "Persistent TCP" and "Pseudo-Web". In "Persistent TCP" the source sends an infinitely large file, and sends packets as allowed by the window based TCP flow control. In "Pseudo-Web" we have used the Bestavros and Crovella [7] data to build a model of how a web surfer might behave. The model is described in Section 5.

For one direction of a TCP connection we sometimes talk about the "entry edge-router" and the "exit edge-router", which are of course the edge-router at entry to and exit from the ATM network. When the direction is not specified, it is understood to be the direction of the data-packets.

In the simulation, if the VC carrying the cells is an ABR VC, there is a "segmentation buffer" and a Virtual Source (VS) at the segmentation point, and a Virtual Destination (VD) at the reassembly point, where VS and VD are associated with the same TCP connection. The VS and VD maintain, in cooperation with the ATM network between them, an "Allowed Cell Rate" (ACR), which is the rate at which the VS is allowed to send cells into the VC. The ACR is a dynamically changing entity. This explains the need for a segmentation buffer: the packet-source reacts only to the returning acknowledgements in deciding when to send more packets. The packet-source does not know the ACR (or even that an ACR exists). The segmentation buffer is used to store cells from already arrived packets that have not been allowed into the VC (into the ATM network) yet. The VS takes cells out of the segmentation buffer according to the ACR, and puts them into the appropriate output (cell) port buffer of the edge-router. In that output port buffer the cell competes for bandwidth with cells from other VCs. Presently, cell output ports in our simulation (in edge-routers as well as in cell-switches) have up to five parallel buffers that can be used to give non–preemptive priorities (see Section 6).

Throughout this paper we will avoid mentioning "sources" and "destinations" without specifying "packet" or "cell" (where "virtual" is understood to imply "cell").

The simulation also includes implementations of UBR (Unspecified Bit Rate), VBR (Variable Bit Rate) and CBR (Constant Bit Rate) VCs. Of both UBR and VBR we have two versions in our simulation: "shaped" and "unshaped". ABR VCs were described in [2]. Brief descriptions of UBR VCs, VBR VCs and CBR VCs will be found in Section 4.2 of this paper.

Figure 1 shows a picture of a single TCP connection over ABR: TCP/IP packets are generated in a packet host (H1) and are sent over a packet link to the entry edge-router. There the packets are segmented and the resulting cells are put in the segmentation buffer of the ABR VC carrying the cells of the TCP/IP connection. The VS (Virtual Source) takes those cells out of the segmentation buffer at rate ACR (Allowed Cell Rate) and puts them into the appropriate cell output port of the edge-router. In that cell output port, the cells compete with cells of other VCs (ABR, UBR, VBR, and CBR). The cells are then transmitted to an ATM network, which in Figure 1 is a single switch. After competition with other cells in the cell buffers in the ATM network, the cells arrive in the exit edge-router, where reassembly occurs. If reassembly is successful, the packet goes to a packet output port of the edge-router, where it competes with packets of other packet connections, and ultimately arrives at the packet destination, packet host H1*.

Figure 1 is a very simple example. The ATM network may contain many switches, and there may be complicated networks of packet routers between the packet hosts and the edge-routers. The simulation tool also allows for purely packet-based networks consisting only of hosts and routers.

4 ATM VCs

The ATM draft standard describes many types of VCs: ABR (Available Bit Rate), UBR (Unspecified Bit Rate), VBR (Variable Bit Rate) and CBR (Constant Bit Rate) For details the reader should see [1] or [2]. The following two subsections give short descriptions suitable for a first reading of this document. The description of ABR VCs has more detail than that of the other VCs.

4.1 ABR VCs

The ATM Tutorial [2] contains a description of how, for ABR VCs, the network (the sequence of ATM switches the VC traverses) sets the ACR. To be more exact: the rules given by the ATM Forum are described, and since the ATM Forum left the "switch behavior" implementation specific, a specific choice for switch behavior is described.

There are two versions of ABR: ABR ER (Explicit Rate) and ABR EFCI (Explicit Forward Congestion Indicator). Both are implemented in the simulation, and both are described in [2].

Both ER and EFCI rely on "RM cells" (Resource Management Cells) to get the necessary information from the switches, or from the virtual destination, to the VS (Virtual Source) so that the VS can update the ACR. The flow of RM cells is depicted in Figure 2.

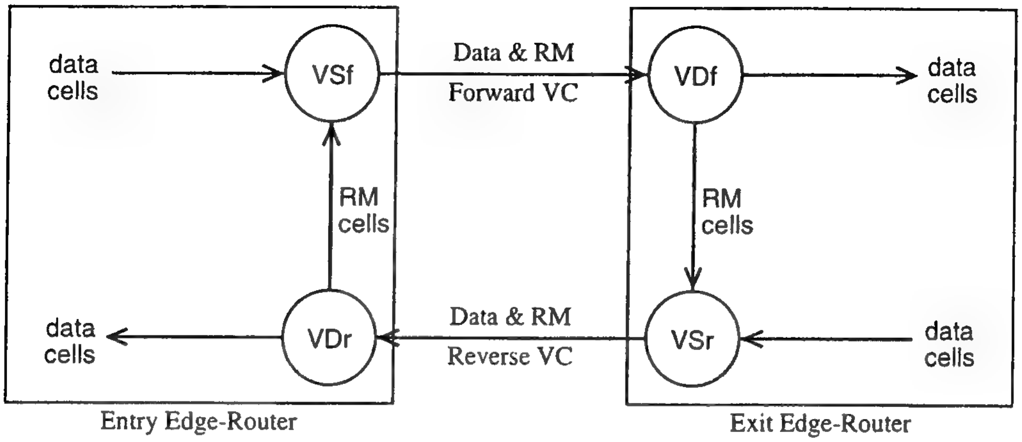

Figure 2: The flow of RM cells

RM cells are created in the VS (say of the forward VC), and pass through the switches (collecting information) to the VD (Virtual Destination). The Virtual Destination of the forward VC may add more information to the RM cell before handing it over to the VS of the backward path, which sends it to the VD of the backward path, where it is handed over to its point of origin. There the collected information is used to reset the ACR. Details can be found in [2].

4.2 UBR, VBR and CBR

The simplest ATM service category is UBR (Unspecified Bit Rate). This is a "Best Effort" mechanism. In (unshaped) UBR packets arriving at the edge-router are segmented into cells and then immediately put into the output buffer. Hence, unshaped UBR VCs do not have a VS or VD, a segmentation buffer or an ACR. We also have a variation on this theme in our simulation: shaped UBR. Shaped UBR VCs do have a VS and segmentation buffer and an ACR, but the ACR is not determined through feedback from the network. In our simulation, the ACR of a shaped UBR VC is fixed and determined at setup of the VC. In the future we may add the ability to have the VS "infer" a rate at which the true source (the IP host) is sending, and set the ACR to slightly higher than that rate.

Another (conceptually) very simple service category are the CBR VCs. These are VCs where the source sends cells (not packets) and sends them at a perfectly constant rate (or, possibly, with always a certain minimal spacing between cells).

The last service category is VBR (Variable Bit rate). In this service category, the VC has been assigned at setup a "sustainable cell rate" and a "maximal burst size". Edge Routers and Switches have the option of using a leaky bucket to monitor the behavior of the VC. If a cell violates the service contract, it can either be dropped or declared (and marked) "out of rate". Cells marked "out of rate" can freely be dropped when there is congestion in the network. We have both "shaped" and "unshaped" VBR VCs. "Unshaped" VBR VCs do not have a segmentation buffer: when a packet arrives at the Edge Roüter it is segmented in into cells and the resulting cells are immediately put in the appropriate output buffer. For unshaped VBR it is strictly up to the "true" source whether the cell stream stays within the envelope of sustainable cell rate and maximum burst size. "Shaped" VBR VCs have a segmentation buffer, and if necessary cells are delayed to stay within the "traffic envelope".

Following the ATM Forum [1] we have VBR-rt (real time) VCs as well as VBR-nrt (non real time) VCs implemented in our simulation, both of them "shaped" as well as "unshaped". In our simulation the only difference between 'rt" and "nrt" is that VBR-rt VCs usually have a higher priority in the cell output ports in Edge Routers and in switches.

In actual use we foresee that VBR-rt VCs almost certainly would not be "shaped", while VBR-nrt VCs probably would be. This makes it somewhat questionable whether VBR-nrt VCs make sense at all: use ABR VCs or shaped UBR instead! VBR VCs are implemented in the simulation but are not discussed further in this report.

5 Source Models

As explained before, for TCP sources our simulation has many source models. Many of these are described in [4], the Manual that goes with the simulation tool we delivered to Rome laboratories. One model called "Persistent TCP" represents the situation where a TCP source transmits a single "Infinitely Large" file, see below. Other models are described in terms of think–times and file sizes: the source "thinks" for a certain amount of time and then sends a file of a certain number of bytes. Among the models are some that are popular in analytic studies and easy to generate, namely with exponentially distributed think–times between files and either exponentially distributed or deterministic file sizes. The more important models are based on actual measurements, to be precise on the "Bestavros and Crovella data" ([7]: a Boston University study of Web access by students). In Web traffic it actually is the destination which "thinks" and then asks for a new file to be transmitted. The source models used in the runs discussed later in this paper are:

- Persistent TCP
- Pseudo-Web

In the Persistent TCP model, the source behaves as if it is FTP-ing an infinitely large file. Every time it gets the ability to do so, according to received acknowledgements and congestion window, it sends a packet.

In the Pseudo-Web model, a source goes through a think time of which the length (in seconds) has a given distribution (see below) and then sends a file of which the size (the number of bytes) follows a second given distribution (see below). Once the last byte of a file has been acknowledged, the source resets its Congestion Window *cwnd* to one MSS (Maximum Segment Size) and its Threshold *ssthresh* to 64 Kbytes (65535 bytes). When the next file is ready to be transported (one think time later) it starts in "slow start". The simulation does not implement the SYN and FIN packets of real TCP (or the routing done by the IP layer – routes are fixed at startup time). For the purposes of this project, we consider these acceptable simplifications.

Because the simulation tool is written in C++, it is relatively easy to add source models, for example Internet Telephony or dependencies between think times and file sizes. In the software, there are different "TCP" modules and "Application Modules". The source model resides in the application module, which mimics the way real applications hand a byte stream to the TCP software.

Think time distributions and file size distributions were obtained from a study of the Bestavros and Crovella data [7]. The Bestavros and Crovella data were obtained by monitoring the behavior of students at Boston University, who were given free access to the Web.

For file sizes in the Bestavros and Crovella data we found (this work was done mostly by our colleague Arnold Neidhardt at Bellcore) a simple distribution (F denotes file size, in bytes):

$$P\{F > f\} = (1 + (\frac{f}{\mu})^{\alpha})^{-1}, \text{ where} \tag{5.1}$$

μ is the median,

$\alpha > 1$, and

$\mu \times \alpha^{-1} \times \Gamma(\frac{1}{\alpha}) \times \Gamma(1 - \frac{1}{\alpha})$ is the mean.

If $0 < \alpha \leq 1$, Equation 5.1 is still a probability distribution, but that distribution has no first moment.

For the Bestavros and Crovella data we found

$$\mu = 219, \ \alpha = 1.151 \, .$$

Hence, the distribution (5.1) has a finite mean but infinite variance.

The mean of the distribution (5.1) is 14,954.22 bytes.

For the "think times" T (in seconds) we found the distribution

$$P\{T > t\} = p_h(1 + (\frac{t}{\mu_h})^{\alpha_h})^{-1} + p_l(1 + (\frac{t}{\mu_l})^{\alpha_l})^{-1}, \tag{5.2}$$

where

$p_h + p_l = 1$, i.e. a mixture of two distributions as in (5.1), and

$$p_h = 0.495, \ p_l = .505$$

$$\mu_h = 10, \ \alpha_h = 1.24,$$

$$\mu_l = 0.245, \ \alpha_l = 3.25.$$

The distribution in (5.2) is a mixture of two distributions, one ("h" for high) long tailed and one ("ℓ" for low) with a finite variance and even a finite third moment. It is attractive to believe that the "ℓ" think times occur when the computer (actually: the browser) chooses the next file to be downloaded while the "h" think times occur when it is the human customer who makes the decision. A further discussion of when "think times" are generated by humans and when they are generated by the computer needs understanding of how browsers work, this would lead us too far afield.

The mean of the distribution (5.2) is 21.83705 seconds. The two distributions that make up the mixture have expected values $E[T_h] = 43.79$ seconds, and $E[T_\ell] = 0.2877$ seconds.

While it is not reasonable to expect that file sizes and think times in a military environment are the same as those requested from the web by students at Boston University, the general shape we see in (5.1) seems typical of just about all environments: the great majority of files transported are quite small (a median of 2190 bytes), but there are some quite large files, and those are so large that actually the majority of all bytes travel in large files.

All simulation results reported on in this paper are obtained with sources that are either persistent or follow some modification of the "pseudo-web" model (see below).

One of the results of interest is the response time of a transaction. We have taken the point of view that the response time is the time from sending the first packet (or first byte) until the last packet (or last byte) of the file is acknowledged. Thus, we are interested in the relationship between file size and response time. For very large files it is impractical to measure the actual response. Instead, we measure the throughput rate (file size divided by response time) of a "very large" file. In fact, we did this for persistent connections, where the rate is the number of of bytes transported in the first (say) 50 seconds, divided by 50. The predicted response time of other "large" (but finite) files then is obtained by dividing the actual file size by throughput rate of persistent files. For non-infinite files we usually restrict the size to an interesting range.

We wanted to study the response times just alluded to in routes that contain a congested link. In order to achieve congestion with the very long think times as in (5.2) we would have needed an extremely large number of sources. To achieve congestion with a much smaller number of sources, but large enough that in our judgement the number of "simultaneously active" connections at any point in time still is sufficiently random, we often decrease the sizes of think times by a moderate factor. This does not substantially change the character of the traffic, while greatly reducing the simulation overhead.

For similar reasons we often chose a file size distribution different from the one in (5.1). Details are given in the next subsection.

5.1 Medium and Heavy Pseudo-Web Sources

In most runs (other than with the original "Pseudo–Web" model) we used a distribution we call "Pseudo-Web Medium".

"Pseudo–Web Medium" is obtained from "Pseudo–Web" by two modifications:

- A major decrease in the lengths of think times
- A minor decrease in the file sizes

As explained earlier, with the Pseudo-Web model in its original form, many sources are needed to cause congestion in a link of considerable bandwidth (for example DS3). While the simulation tool can easily handle this, we often found it more convenient to decrease the think times between the files of a source. In one model, which we call "Heavy", we only used the "ℓ" distribution in (5.2). This however makes the number of simultaneously active connections to close to deterministic. In the "Pseudo–Web Medium" model we also decrease the think times, but less drastically.

The think times in "Pseudo–Web Medium" are obtained as follows: first draw from the distribution in (5.2), divide the result by 10 (to get a mean of 2.18 seconds), but then resample if the result is below 40 msec. The result is a mean think time (empirically) of 3.1 seconds.

With this choice, in most runs, "think times" are still large compared with the times it takes to transport a file. Hence, at any point in time the number of simultaneously active files has about the same distribution as it would have with more sources, longer think times, and about the same load.

With lower actual bandwidth per active connection, either because of lower bandwidth access channels or because of more congestion, the time to transport a file becomes longer and at some point has the same order of magnitude as the think times. At that point the use of "Pseudo–Web Medium" leads to a number of simultaneously active connections that no longer is approximately Poisson. In some of our runs (with DS0 access links instead of T1 or higher bandwidth access links) we may be close to that situation.

We do not expect the results to depend strongly on the actual shape of the think time distribution.

As explained earlier, in many of our simulations we use a workload consisting of a few "persistent" TCP connections, plus many that are either "Pseudo-Web" or "Pseudo-Web Medium". With the file size distribution we use, there are only a few very large files. While such a file is being transported it is as if there is one more persistent connection. Thus, having a random number of files that stay around for a long time makes the results more random.

For this reason we restrict, in the "Pseudo–Web Medium" model, files to having at most 0.5 Mbytes. If the distribution (5.1) produced a file of more than 0.5 Mbytes, we resampled.

Also, we decided not to have files of fewer than 20 bytes. If sampling from (5.1) produced a file of less than 20 bytes we replaced it with a file of 20 bytes. (We did not resample.)

With this method of sampling we observed that about 45 percent of the bytes transported were in files of more than 100 Kbytes.

Without this reduction in file sizes, about 93% of all bytes are in files over over 100 Kbytes.

6 Purpose and Use of Priorities

In the simulation, in its current form, there are explicit priorities only in the ATM part of the network. In the IP part, a host can effectively be given a higher priority by giving it an access channel with larger bandwidth.

In the ATM part of the network priorities can be given in three different ways:

- Explicit priorities in the switches
- Differential weights for ABR VCs (ER only)
- Differential sustained rates and burst sizes for VBR VCs

While VBR VCs are implemented in the simulation tool, they are not discussed further in this report.

Explicit priorities are implemented in the cell ports in the Edge Routers and (cell) Switches. Those ports have five parallel queues (numbered 0 to 4), with the lowest number having the highest priority. Cells are assigned according to their type, not according to their VC. We have six types: CBR cells, RM cells, VBR-rt cells, ABR data cells, UBR cells, and VBR-nrt cells. The types can be assigned to priority classes in any arbitrary way. In the runs we report on in this paper we always have the assignment

- **Level 0:** CBR cells.
- **Level 1:** RM cells (ABR RM cells).
- **Level 2:** VBR-rt cells.
- **Level 3:** ABR and UBR data cells.
- **Level 4:** VBR-nrt cells.

The fact that RM cells have priority over ABR data cells means that RM cells can, and often do, overtake data cells of the VC they belong to. Within RM cells and within (for example) ABR data cells sequentiality is maintained.

We could have used the same mechanism (multiple parallel queues with service priority between the queues) to assign priorities to specific VCs. In case of ABR ER ports, with the original max-min method for assigning Explicit Rates, this would not have been effective, because the bandwidth assigned to a VC would not depend on its priority level. Thus we invented, and implemented in the simulation, "Weighted Max-Min". In weighted max-min VCs are given a weight which is an (increasing) function of their importance.

There are at least two reasons one might give differential weights to different ABR VCs:

- To achieve desired throughput ratios
- To achieve preferential treatment

We will discuss these intents in the next two subsections.

6.1 Throughput Ratios

This intent is most obvious in a commercial environment, but the need to consider it may also occur in other environments. Suppose a number of companies are sharing a link. They are likely to want to make sure that every company gets its fair share. One way to achieve this is by giving each of the companies an ABR VC, with a weight equal to its fair share.

This works only if the bottleneck output port implements ABR ER. In that case, it is clear that as long as all companies always (or virtually always) have data to send, they will get bandwidths in almost perfectly the desired ratios. This is "theoretically obvious" from the descriptions in [2] and is borne out by our simulation (see Section 7.3).

A problem arises when not all "companies" always fill their quota. In that case it is desirable that the left over bandwidth is distributed equitably over the other VCs, as it would be under "Weighted Fair Queueing" (WFQ, see [13], [14], [15]). The current implementation only partially reaches that objective, so it could be called a form of "poor man's WFQ". In the current implementation the switch takes away the allotment of an ABR VC if it completely stops sending for a period of between 100 and 200 msec (or longer). If it decreases its rate, but sends a forward RM cell at least once every 100 msec, it keeps as ACR the minimum of what it "requests" and what is allotted to it by the weights alone.

The problem arises when a VC (actually, a VS) asks for its "allotted share" or more but does not use it. In that case, the switches set aside the allotted share, and the unused part goes to waste even when other VCs might want it.

This is the situation currently in our simulation. With an implementation where VSs do not ask for more bandwidth than they actually are going to use, this problem (or feature?) would go away. If the Virtual Source also is the true source of the traffic, such a modification should not be hard to implement. If, as in the situation studied in this paper, the Virtual Source and "true source" are not only different but do not know of each other's existence, such a modification requires development.

6.2 Priorities

A second use of weights for ABR VCs is to set priorities. The prototypical situation is that of a VC which usually does not send, but occasionally sends a large file and when this happens needs a fast response time, i.e., a short time until completion of the file transfer and a high bandwidth while the transfer is in progress. Section 7.3 in this paper further discusses use of weights for this purpose.

As long as such VCs are active only rarely, they do not affect other VCs unless they need the bandwidth, but when they request bandwidth they instantaneously are allotted a large amount. Of this large allotment they can of course initially use only the ICR (Initial Cell Rate), but the ACR then grows quickly to what the switches permit.

There can be a somewhat gradual transition from this situation to the situation above when the "important" VC becomes active more often for sustained periods of time. In the intermediate situation there is the risk we already discussed, that the important VC asks for bandwidth and is assigned bandwidth but does not use it.

7 Simulation Studies

This section gives detailed results on the simulations that were run for the study. The descriptions assume familiarity with the introductory material, in particular regarding the various type of ATM Virtual Circuits: UBR, CBR, ABR, and VBR.

In this section we often use the concept of "normalized throughput". This is the throughput in real data, excluding TCP and IP packet headers and Cell Headers and RM cells, as fraction of theoretically possible. Because we have TCP packets with 536 bytes per packet plus 40 byte IP and TCP headers, for a total of 576 bytes per packet, and because every ATM cell has a 5 byte header, the theoretically possible normalized throughput is at most 84.28%. If ABR is used, so that there are RM cells, the theoretically possible total normalized throughput is even lower. When one out of every 32 cells is a forward RM cell the highest possible total normalized throughput would be 81.6%. Backward RM cells of the reverse direction VC reduce the total normalized throughput even further. Actual total normalized throughputs are even lower because of a variety of reasons, such as sources not utilizing the bandwidth set aside for them.

7.1 A problem with ABR-ER: Fluctuating VCs

We will see that in most situations ABR-ER works reasonable well or better, often even extremely well. We found one situation where use of ABR-ER causes performance degradation. This is the situation where ABR VCs frequently alternate between having cells to send and being inactive. We call such VCs "Fluctuating VCs". The ABR scheme used in our simulations does not have a mechanism that allows the Virtual Sources to tell the switches it is no longer using its ACR. The result is that when a Virtual Source runs out of cells to send, the switches serving the VC keep the reserved bandwidth in existence for at least 100 msec and at most about 200 msec. This phenomena was predicted in [2] and is studied here.

The network used to quantify this problem is given in Figure 3. In this network the bottleneck link is a DS3 link that transports cells. The links between Edge Routers and Switches are DS1 or DS3 links, also carrying cells. The links between Hosts and Edge Routers are DS1 or DS3 links and carry packets. The two links on an Edge Router always

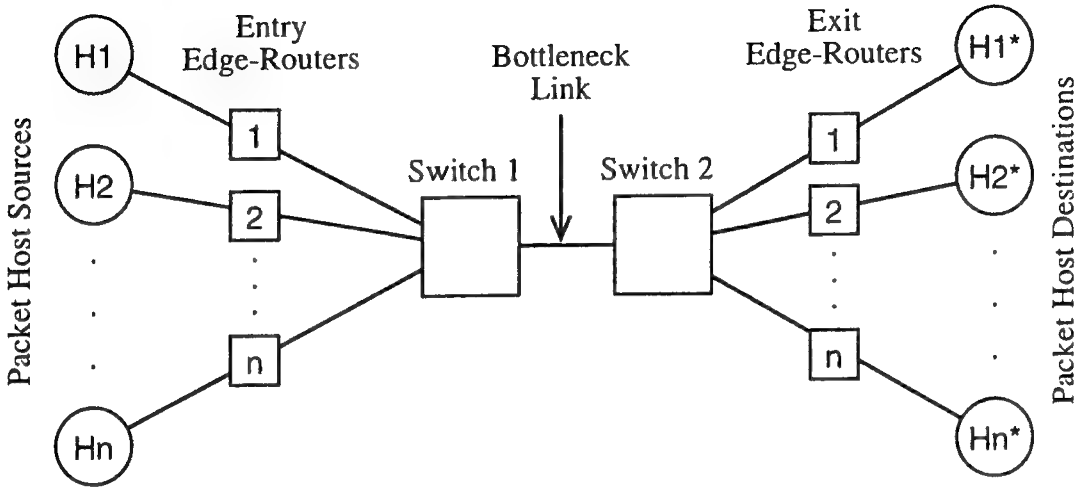

Figure 3: Generic Two Switch Network

have the same bandwidth. A host is called a "DS1" host if it is connected to its Edge Router through a DS1 link, and is of course called a DS3 host if it is connected to its Edge Router through a DS3 link. We did two series of experiments. In the first series, we have three "Persistent TCP" sources each connected to its Edge Router through a DS3 link, plus a number of "Pseudo–Web Medium" sources, also each connected to its Edge Router through a DS3 link.

Figure 4 gives total normalized throughput through bottleneck link, as well as the parts due to the three persistent sources and the (up to 200) DS3 sources. We see that total normalized throughput decreases when the number of fluctuating VCs increases: an undesirable situation!

Figure 5 gives similar results for the situation where the fluctuating VCs go through a DS1 link before reaching the bottleneck. The result is that each such VC has an ACR of at most DS1. Hence, the bandwidth set aside but not used is at most DS1 and the deterioration is less severe.

A conclusion to be drawn is that "Fluctuating" VCs are undesirable when there are

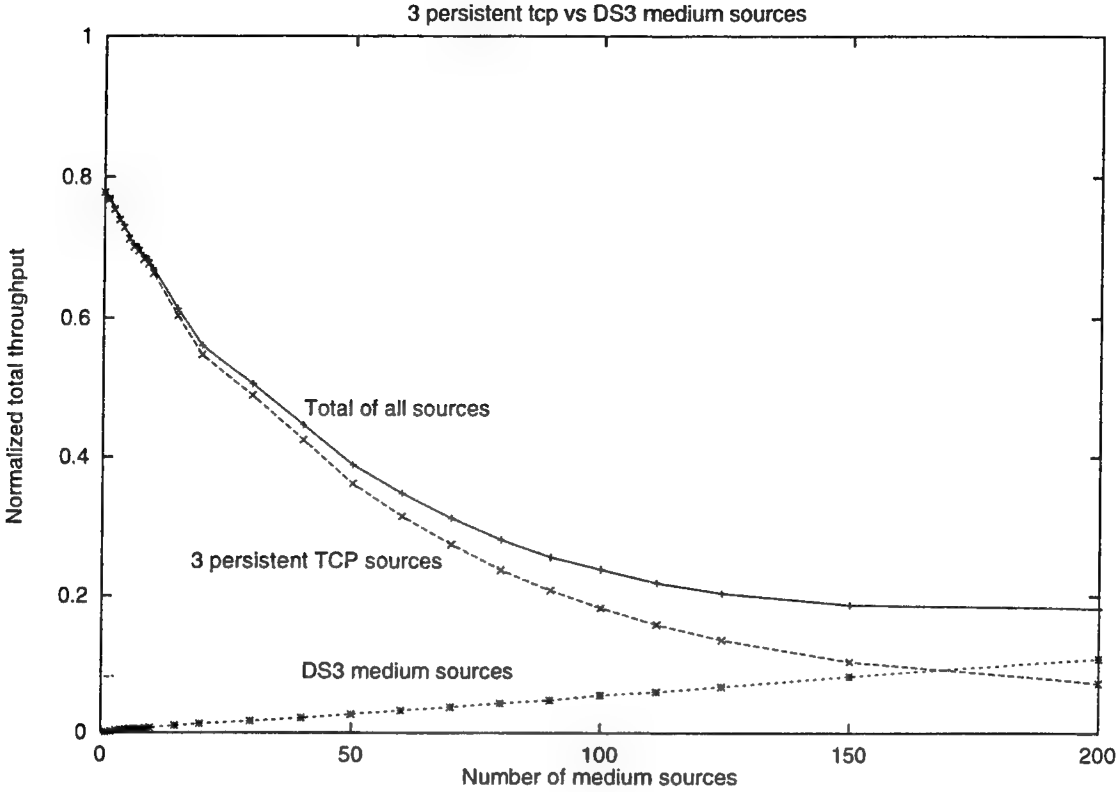

Figure 4: Fluctuating DS3 VCs

ABR-ER capable switches in the path. The problem can be partially eliminated by enabling Virtual Sources to send an RM cell "telling" the switches in the path that it is voluntarily decreasing its ACR. This issue may need further research.

7.2 The Effect of Cross Traffic

In [6], [5] it is shown that the Congestion Window of a persistent TCP connection of which the Maximal Window is large typically fluctuates around $p^{-1/2}$ Maximal Segment Sizes (MSSs), where p is the drop probability. Hence, the throughput rate of such a connection

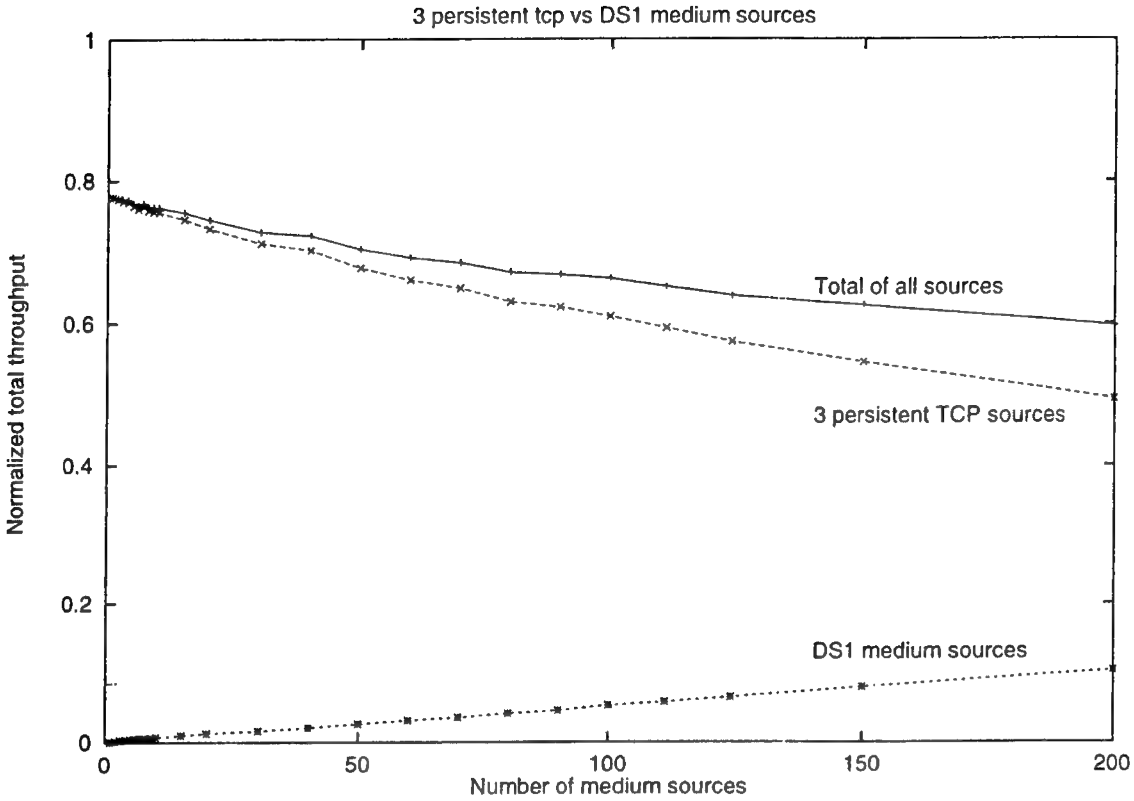

Figure 5: Fluctuating DS1 VCs

typically is in the order of

$$\frac{1}{RTT \times \sqrt{p}} \tag{7.1}$$

(MSSs per second), where RTT is the round trip time.

In a network with multiple congested gateways as presented in Figure 6, if all switches have the same bottleneck line speed and about the same loss probability and queueing delay, and all links have the same propagation delay, a connection traversing n gateways has about n times the delay, and n times the loss probability, of a connection traversing only one gateway.

In the specific situation of a single connection traversing n gateways, with in every gateway competition of a single cross connection traversing only that gateway, the throughput of the connection traversing n gateways is in the order of $1/(1+n\sqrt{n}) \sim 1/(n\sqrt{n})$ of the bottleneck bandwidth.

Similar problems were studied by Sally Floyd in [20].

This is the situation where there is no ATM or ABR and loss is uncorrelated (between different packets in the same stream).

We decided to investigate similar phenomena in networks where the intermediate gateways are ATM switches rather than routers. The effect of different ATM VC types (UBR and ABR using either ER or EFCI flow controls) are examined.

The networks we study in this section all have the form shown in Figure 6. All the flows are persistent TCP connections with very large windows, assuring that congestion will eventually occur. The flow we are studying, "Flow 0," goes from the leftmost host to the rightmost. The "cross" flows, 1, 2, 3, etc., each share one link with Flow 0. Note that although one cross flow enters and another leaves in intermediate switches, these flows do not interact with each other because they do not share any output ports. This representative network stands for a family of networks that we have studied with between one and nine cross flows.

All the links in the network are DS3s, and each link has a propagation delay of 1 millisecond. Thus, the round trip time (exclusive of queuing delays) for the cross flows is 10 msec, while for Flow 0 it is $(8+2n)$ msec, where n is the number of cross flows. All flows are persistent, meaning that they continue to send traffic indefinitely at the highest rate allowed by their current windows. As in most of our simulations, we use delayed ACKs with our version of TCP-Reno protocol and the default for dropping cells and packets (tail drop).

Figure 7 shows the throughput for Flow 0 as the number of competing connections goes from one to nine. All of the points in the figure are derived from simulation runs

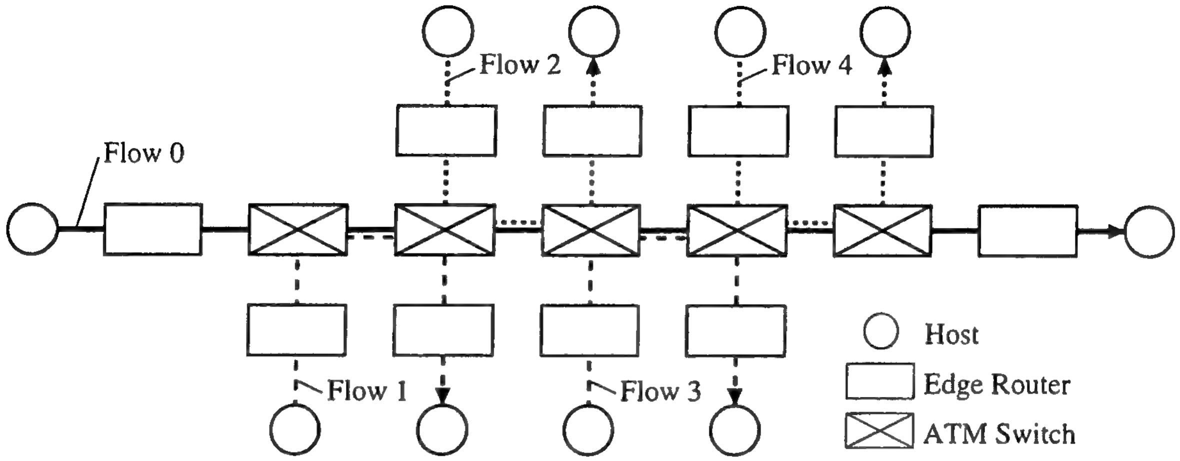

Figure 6: Typical Network with Cross Traffic

of 120 seconds, measuring the total throughput from the 20 second mark until the end of the simulation. The data points for the UBR and EFCI points have a good deal of variance, so the plotted points should not be taken as precise values.

7.2.1 UBR

ATM flows using UBR (Unspecified Bit Rate) are not rate controlled. Thus, the network using UBR is comparable to the Floyd network based on routers, which also provide no flow control.

As expected, Flow 0 gets half of the available throughput with only a single cross flow. The throughput value for one cross connection is about 0.42 rather than 0.5 because the plotted values account only for payload traffic. The overhead of five bytes for every 53 byte ATM cell plus, to a lesser degree, the 20 bytes of IP header per packet account for essentially all of the remaining bandwidth.

As the number of cross connections increase from one to nine, the throughput for Flow 0 decreases precipitously. This is because each TCP packet suffers an independent chance of being dropped or damaged (one cell dropped will prohibit the packet from being delivered) at each switch for which there is cross traffic.

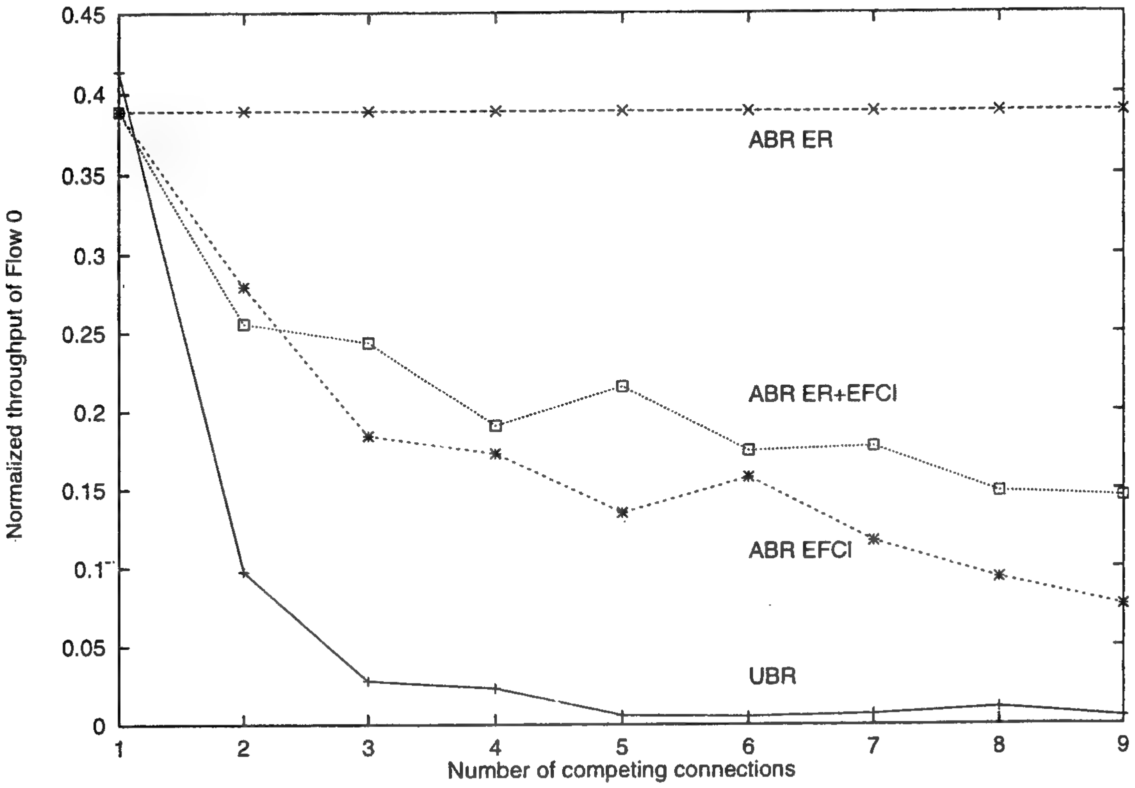

Figure 7: Throughput of Flow 0

7.2.2 ABR using Explicit Rate

The ABR flow control method for ATM is designed to share the available bandwidth fairly amongst all actively competing PVCs. In our network, each TCP connection is carried over a separate PVC, so if sharing works properly, each connection, including Flow 0, should get half the bandwidth. There are two methods used to control the flow at the source. The Explicit Rate (ER) method passes a value through the network as part of an RM cell. Initially, the ER value is set to the current Allowed Cell Rate (ACR) of the connection. At each switch, this rate may be reduced to provide for sharing the available bandwidth fairly. When the RM cell returns to the source, the flow rate is set to no more than the contained ER value. As can be seen from Figure 7, this method works extremely well, and Flow 0 gets fully half of the available bandwidth, independent of the number of competing cross connections.

Notice that while ABR ER provide a throughput which is essentially independent of the number of cross connections, the value for ABR ER is actually slightly less than that for UBR when there is exactly one cross connection. This is explained by the additional overhead (of about 3%) incurred by ABR in sending RM cells (roughly every 32nd cell will be an RM cell).

7.2.3 ABR with EFCI

Although the ER method appears to work very well, most ATM switches today that provide ABR use another, less sophisticated method called Explicit Forward Congestion Indicator (EFCI), see [2]. If a switch finds that the output queue for a port is filled above a certain level, the EFCI bit is set in the cell header. (There is a provision to provide for hysteresis, so that EFCI bits are set from a point when a certain high level is reached and continue to be set until another, lower level is reached. In practice, these two levels are normally set to be equal. In our simulations, we have set the value for setting the EFCI bit to 75% of the queue length.)

The destination end of an ATM connection keeps track of the EFCI bits on the incoming data cells. When an RM cell arrives and is returned back to the source, the CI bit in the RM cell is set if at least one data cell arrived since the previous RM cell, and also the EFCI bit of the last arriving data cell was set. When the RM cell returns to the source with its CI bit set, the source reduces its previous Allowed Cell Rate (ACR) to ACR times (1 - RDF) (RDF is the Rate Decrease Factor). In the runs we are reporting on RDF is always equal to 1/16. The ACR is never decreased below MCR (Minimal Cell Rate). When the RM cell returns to the source with its CI bit not set the ACR is increased by (at most) the PCR times RIF (Peak Cell Rate times Rate Increase Factor). In the runs we are reporting on RIF always equals 1/32. ACR never increases above PCR or above the returning ER (Explicit Rate) value. A more detailed discussion of the algorithm for setting the new ACR is given in [2].

The EFCI control clearly has the property that a flow going through several congestion points has a much higher probability of being reduced than a competing cross flow that only goes through a single congestion point. Figure 7 shows that there is a trend in reducing the throughput of Flow 0 as a function of the number of cross flows that is comparable to that of the UBR case. Although an improvement over UBR, from a fairness point of view the ABR EFCI control is clearly inferior to ABR ER.

7.2.4 ABR with a Mixture of ER and EFCI Switches

In any given network, there can be some ABR switches that use the ER method and others that use EFCI. The standard allows for this, and provides an algorithm for computing the new ACR depending on both the ER and the status of the CI bit in a returning RM cell. Essentially, both controls are applied, so the new ACR is the minimum that would be set using either method alone. See [2] for details.

The curve marked "ER+EFCI" in Figure 7 shows the throughput when half the switches are EFCI (if s, the number of switches is odd, then there are $\lceil s/2 \rceil$ ER switches and $\lfloor s/2 \rfloor$ EFCI switches). As expected, and verified by experiment, the throughput values for Flow 0 do not depend on the ordering of the intervening ABR switches.

As we have seen, the ER based switches effectively cooperate to fairly share the bandwidth, but the EFCI switches interfere with the through traffic and "unfairly" restrict

Flow 0. In a mixed network, we actually see a combination of both effects. Figure 8 illustrates the combined effect on the throughput of Flow 0 when both ER and EFCI switches are traversed.

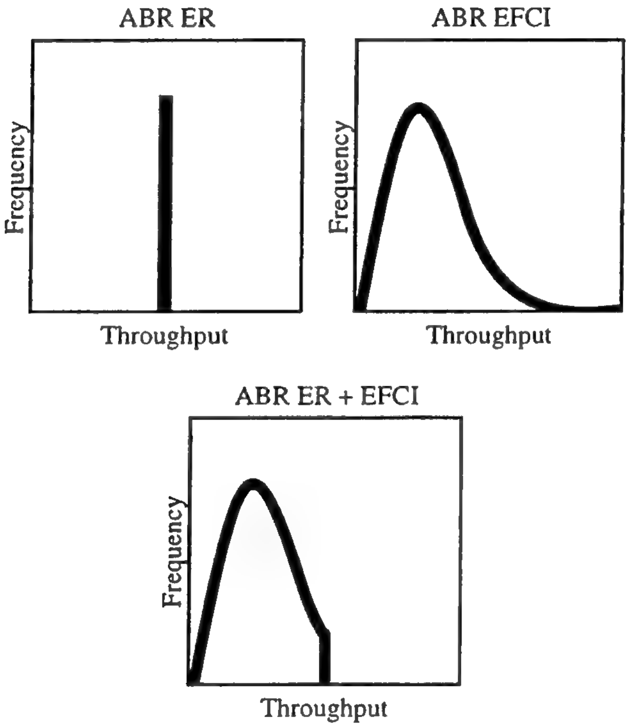

Figure 8: Combined Effect of ER and EFCI

Note that if there is a single ER switch in the path, then the through flow will be constrained to take no more than half the available bandwidth, even if the EFCI control would allow it to take more.

Figure 9 adds four additional points to the previous throughput graph. The point labeled "FFRFF" describes a network with four EFCI switches and one ER switch. We can see that the throughput is slightly less than is obtained with a network of only four EFCI switches (although there is enough noise in the data that this reduction is not

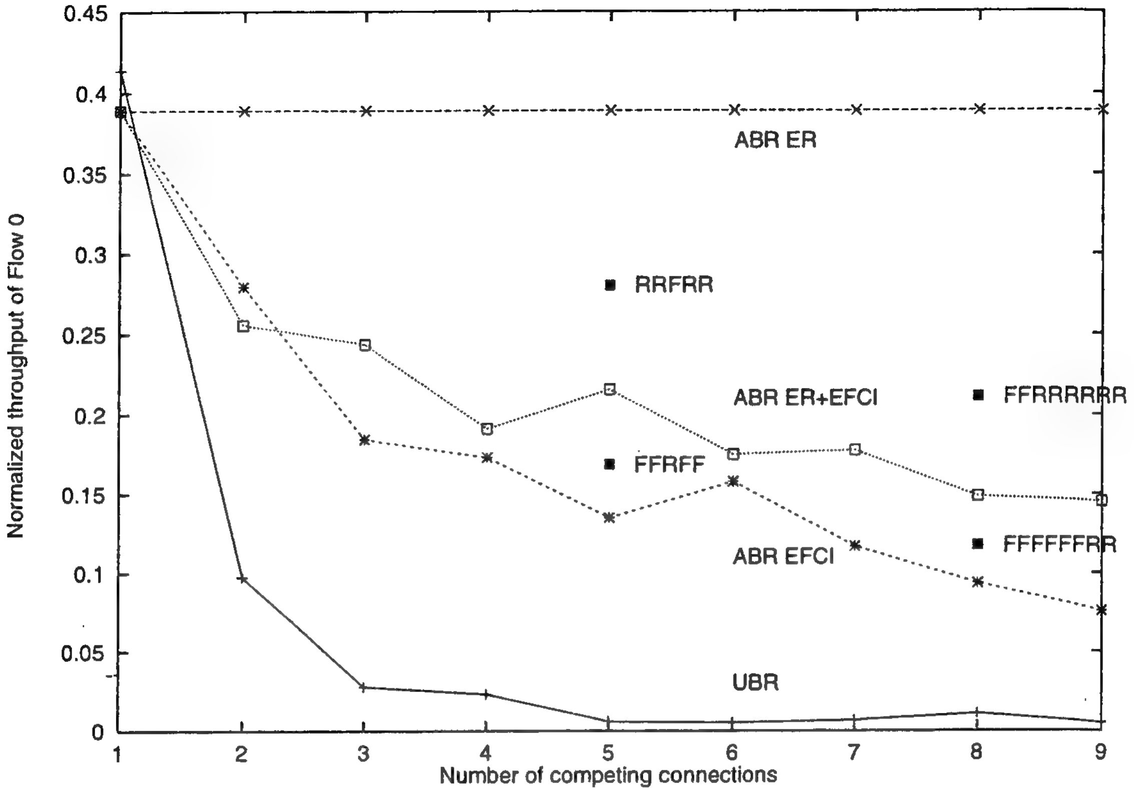

Figure 9: Throughput of Flow 0 with Added Points

significant). Similarly, the point labeled "FFFFFFRR" has six EFCI switches and has lower throughput than the corresponding point with only six EFCI switches.

Adding a single EFCI switch to a network otherwise composed of ER switches will have a significant effect on the throughput of Flow 0. The points labeled "RRFRR" and "FFRRRRRR" show the effect of adding only one or two EFCI switches to a network that otherwise uses only ER switches.

We have seen that ABR based networks consisting of all ER based switches can truly provide fair bandwidth sharing, in particular for non–fluctuating VCs (i.e., either VCs that transport very large files, or VCs that carry data for sufficiently many connections that there always are cells to be sent). We have seen that mixing ER and EFCI leads to deterioration, and that a VC passing through many EFCI switches can encounter "beat-down". Unfortunately, for the near future it is likely that in large networks there will still be a few EFCI switches active, which can delay the promise of ABR.

7.3 Priorities through Weights and through Access Line Speeds

In 6.1 we saw that there are two ways to give a specific connection (VC, or TCP connection including the ATM VC it uses) priority: by giving the VC extra weight, and by giving the TCP connection, including the VC it uses, extra bandwidth in the access network to the backbone network. In this section we investigate how these mechanisms interoperate.

For this investigation we did two series of experiments. In both series, the network used was the one in Figure 3. In both series there are three persistent TCP connections, plus 190 "low priority" TCP "Pseudo Web Medium" connections, plus ten "high priority" TCP "Pseudo Web Medium" connections. Each TCP connection has its own pair of hosts, its own pair of Edge Routers, and its own VC. In all cases the bottleneck link is a DS3 link.

In the first series the 3 persistent connections have weight 1 (i.e. their VCs have weight 1) and have DS3 links (from hosts to Edge Routers and from Edge Routers to Switches), and the 190 "low priority" pseudo–web connections have weight 1 and DS1 links. For the "high priority" pseudo-web connections there are four possibilities:

- all weight 1 and DS1 links (no priority)
- all weight 10 and DS1 links (priority by weight only)
- all weight 1 and DS3 links (priority by access channel only)
- all weight 10 and DS3 links (priority by weight as well as access channel)

Results for the four cases are given in Figure 10. In the lower left plot, the 10 "high priority" VCs have priority neither through weights nor through access line speed. In the upper right plot, they have both kinds of priority. In the lower right plot, priority is given only by weight, and in the upper right plot priority is given only by speed in the access lines. The three lines in each of those figures (one solid, two dotted) give for the 3 persistent TCP connection the relationship between byte number and elapsed time from where byte zero is sent until a given byte is acknowledged. For each of the files transported by the other connection we plot a single point giving elapsed time versus byte number for the last byte, i.e. the file size. Each of those points is the end of a trajectory as given for the persistent sources. For files in the "low priority" TCP connection this end of the trajectory is given by a dot, for files in the "high priority" connections it is given by a cross.

In Figure 10, lower left, there is no difference between low priority connections and high priority connections. In Figure 10 upper right there is maximal difference between the low priority connections and high priority connections. We see that in this series of experiments giving only weight priority has almost no effect. Giving only access line speed priority gives a significant amount of advantage (this makes the files in high priority connection equivalent with the corresponding first part of a persistent connection), and giving both types of priority gives a slightly larger advantage.

In Figure 11 we have a similar series of experiments. The only difference is that now the 3 persistent sources have weight 10.

In this second series giving only one type of priority has only limited effect, while giving both types of priority has significant effect.

The difference is of course that in the first series, in the "base-situation" with no priority, it is low weight of the pseudo-web connections which keeps these VCs from taking more bandwidth from the persistent connections: Only the "weight" constraint of

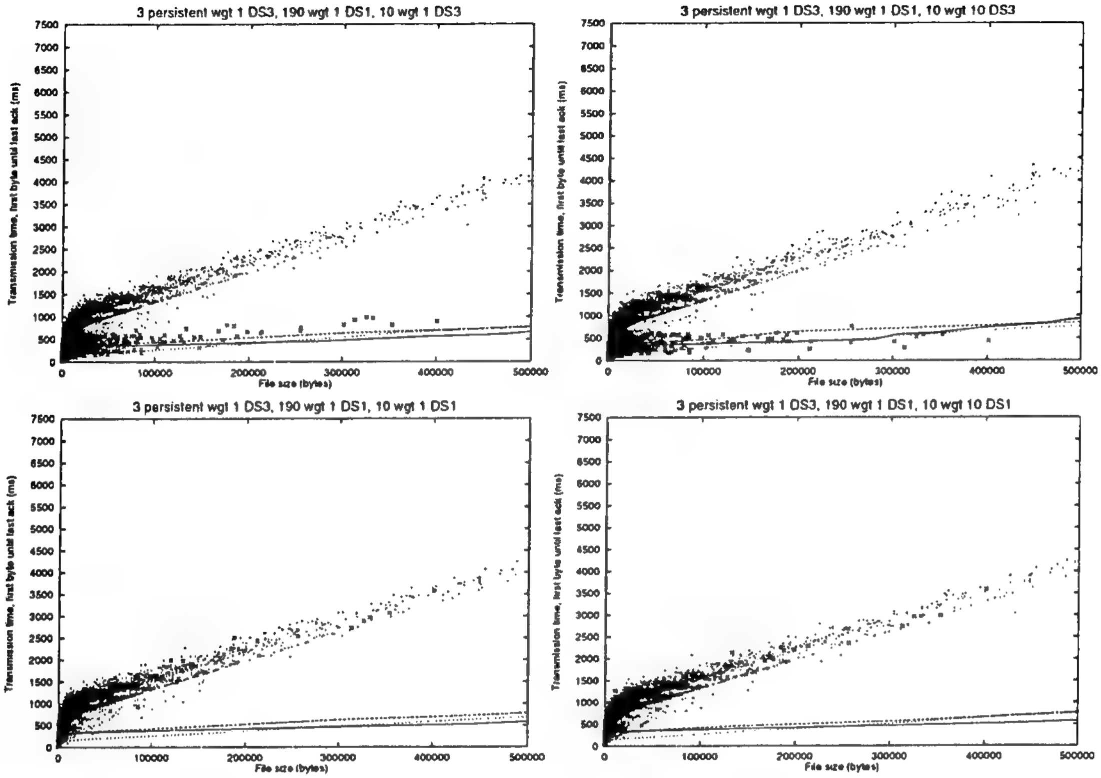

Figure 10: Four combinations of priorities, Persistent T1

the pseudo-web connections is active, and removing it makes a big difference.

In the second series, in the base-situation, bandwidth of the access channels of the pseudo-web connections and weights of the pseudo-web connections both are (essentially) active constraints which keeps these VCs from taking more bandwidth from the persistent connections. Removing just one of the constraints has little effect, but removing both has a major effect.

In the base-situations (lower left plots, in going from the first series to the second, the persistent connections get higher weights and thus higher ACRs. As result, the pseudo-web connections get lower ACRs, and the ACR constraints go from “inactive” to “active”. By increasing the weights of the persistent connections even further, the

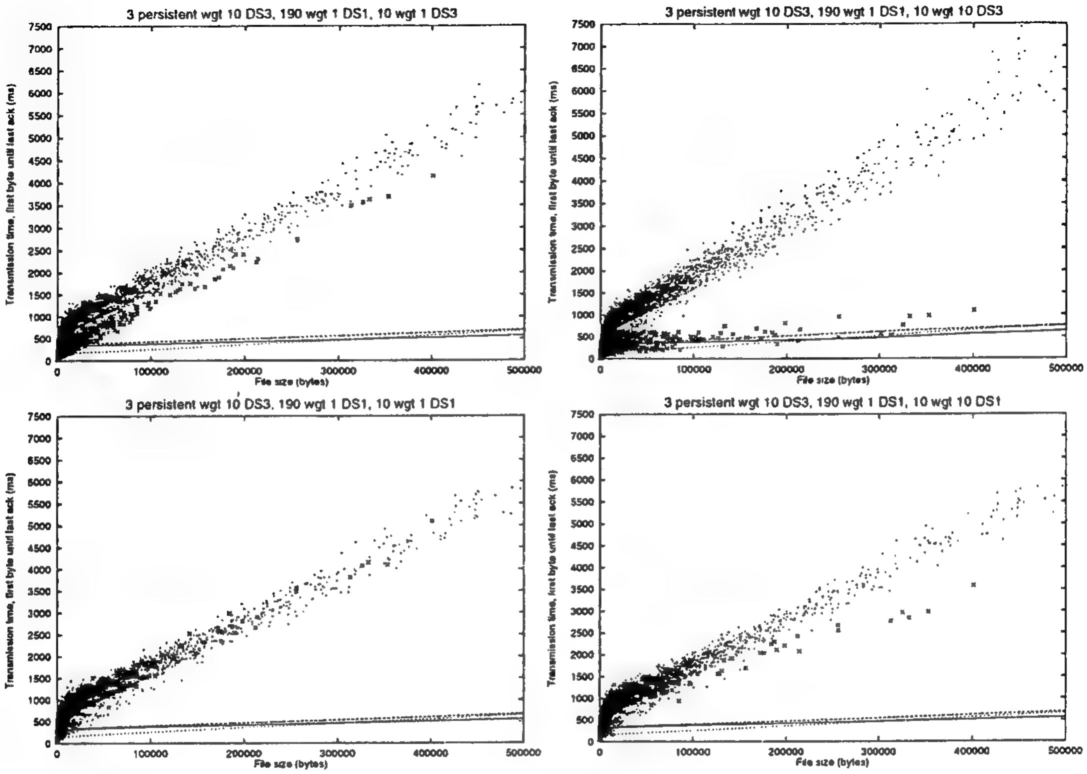

Figure 11: Four combinations of priorities, Persistent T10

bandwidth constraint of the pseudo-web connections would have become inactive.

7.4 The Effect of Random Loss

This study was designed to gain an understanding of how losses on the links in a mixed IP/ATM network affect throughput for sources that simulate traffic that is similar to that generated by web browsing. These are random losses on the links, due to (e.g.) radiation or thermal noise or other physical phenomena, not loss due to congestion. We use the standard two switch, one bottleneck network as illustrated in Figure 3. The delays for all links (IP and ATM) was fixed at 1 msec, so the minimum round trip time for each TCP connection was 6 msec.

In this study all sources follow the "Pseudo Web Medium" model described in section 5.1.

We studied several different combinations of numbers of sources and loss rates. Even with a fairly high loss rate of one cell in one thousand (roughly one packet in 80: high for random loss, not necessarily for congestion loss), the losses had in most cases little effect on file transmission times. This is consistent with the "square root formula", see [5], [6]. This square root formula shows that when a TCP connection transports a large file while the packet loss probability is p, as long as there are no other constraints, the Congestion Window *cwnd* and the throughput will be in the order of magnitude of $p^{-1/2}$ (packets), respectively $p^{-1/2}/RTT$ (packets/sec). Hence, as long as the loss rate due to random loss is small compared with loss rates due to other causes, random loss has little impact. This argument shows that on a link with high bandwidth (more exactly: on a VC with high ACR) a loss rate of one cell in thousand is likely to make a significant impact, but on a DS0 link it makes little difference.

One difference was observed: Without random loss, (with congestion loss only), a TCP connection in our simulations rarely loses one of the first packets of the file it transports. *With* random loss losing one of the first packets gets higher probability. We report on one experiment that illustrates the difference.

In this experiment, the bottleneck link is fed by 40 sources with (weight) priority 1 and using a DS0 feed link. In addition, there were 4 sources with a high weight priority (10) using DS1 feed links.

Figure 12 shows the transmission time results for both experiments side by side. In the left figure, there are no random losses. The points marked as "x"s are from the high priority sources and require dramatically lower transmission time than the low priority sources. In the right figure, a random loss rate of 1 in 1000 has been added, resulting in a noticeable number of files with substantially delayed transmission times. Although it is

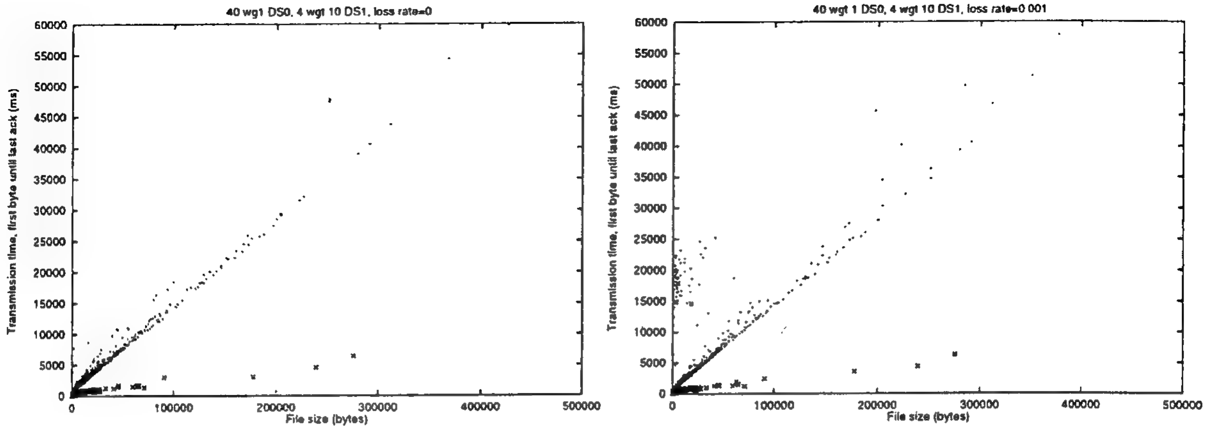

Figure 12: Comparing No Loss and Loss Experiments

difficult to see in the figure, there are some high priority sources in the cluster of delayed files.

The explanation is that when a TCP connection loses one of the first packets of a file, the RTT estimation (actually: the estimation of the Standard Error of the Round Trip Time) has not converged yet and still has the initial high value. If time-out occurs (which is likely if the *cwnd* still is low) the time-out takes a long time (up to about 12 seconds, or even more, see [3]). This is independent of the weight priority of the VC carrying the connection.

We did a few experiments with correlated loss (varying rate of random loss), with (random) loss rates in the order of one in thousand and somewhat higher. We did not find interesting behavior. Since this phenomena is outside the scope of the contract, we did not follow up with higher loss probabilities.

7.5 Propagation Delay

Section 4.1.5.3 of the Statement of Work asks that we investigate the effect of differences in propagation delay in different parts of a network, in particular the effect of delays being in either the cell part or the packet part of the network. The simulation tool makes this easy to do, but we could not find any significant effects caused by varying the location of the delay. TCP throughput is primarily determined by the available bandwidth, round trip time, and congestion caused by competing traffic.

We used the network described in Figure 13 to investigate effects due to the location of the propagation delay. This is a generic two–switch network. The bottleneck is the link between the two switches. There is a variable number of hosts offering pseudo–web medium traffic to the network.

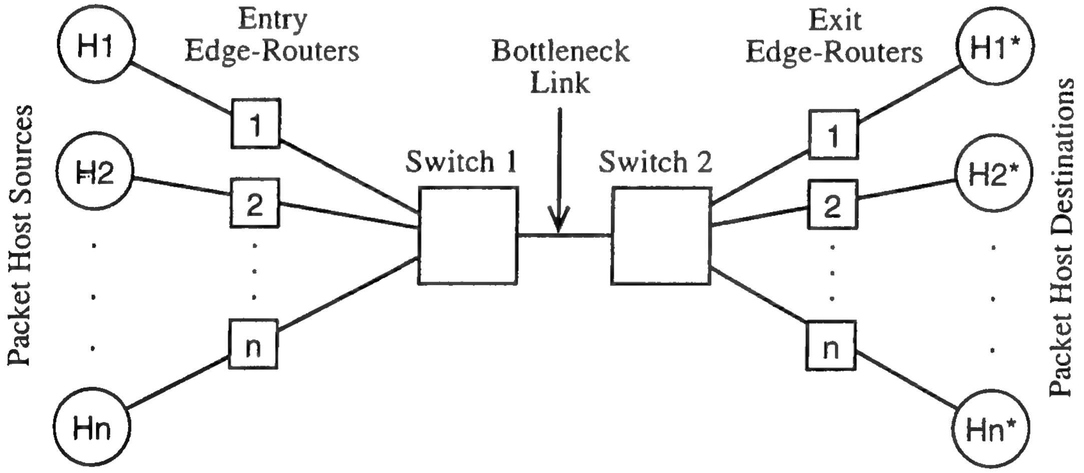

Figure 13: Generic Two Switch Network

All links in this network are DS0 (allowing us to keep the number of hosts to a reasonable number while still providing enough load). The delay on the bottleneck link is kept at 1 msec. To examine the effect of delay in the packet part of the network, we set the delay on the links between the hosts and edge routers to 20 msec and the delay on the ATM links between the edge routers and the switches to 1 msec. We then interchanged these delay choices. For both cases, the round trip time (excluding queueing delay) is 86 msec for all sources.

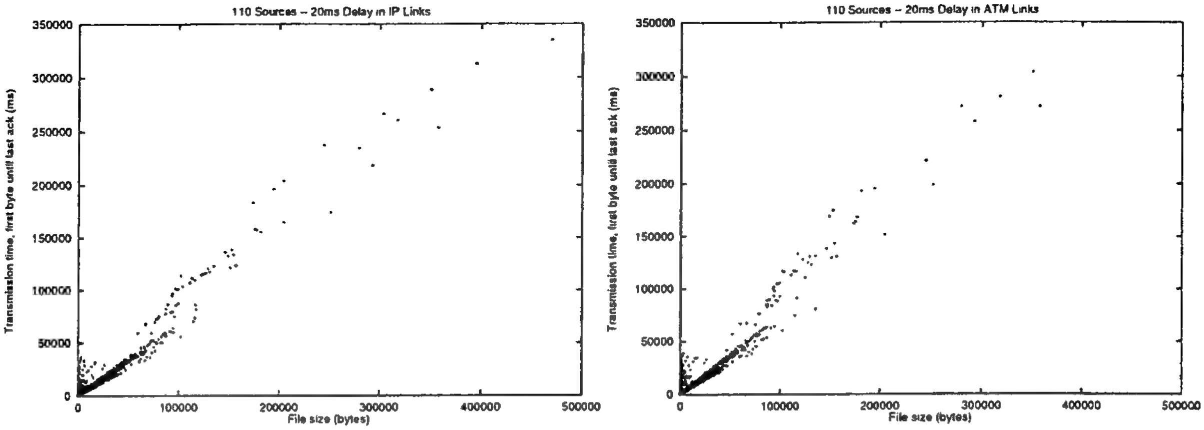

Figure 14: Comparing Delay in Packet and Cell Links

We investigated networks with 33, 44 and 110 sources. The smaller choices do not provide enough load to stress the bottleneck link. Figure 14 show the delays experienced by the files sent by 110 sources. As can be seen in the figure, there is no appreciable difference in the response time experience by users whether the excess delay is in the packet or cell portions of the network. It should be noted that the extreme delays seen here (on the order of 50 seconds for a 100 Kbyte file) are a result primarily of the long round trip time coupled with the overloading of the bottleneck DS0 link. For 33 and 44 sources, the bottleneck does not become congested and the response times are much more reasonable (100 Kbytes are transmitted in about 15 seconds, which is about all we can expect over a DS0 link).

7.6 Buffer Size

This section examines the effect of buffer size choices for networks of the form illustrated in Figure 15. In all cases the packet links have a delay of 10 msec and the cell links have a delay of 1 msec. The minimum total round trip time (RTT) of packets therefore is 44 msec, and the minimum RTT of RM cells is 4 msec. We examined networks with 2, 6 and 15 persistent TCP sources, varying the size of the the segmentation buffers and the switch buffer from 128 cells up to 6144 cells.

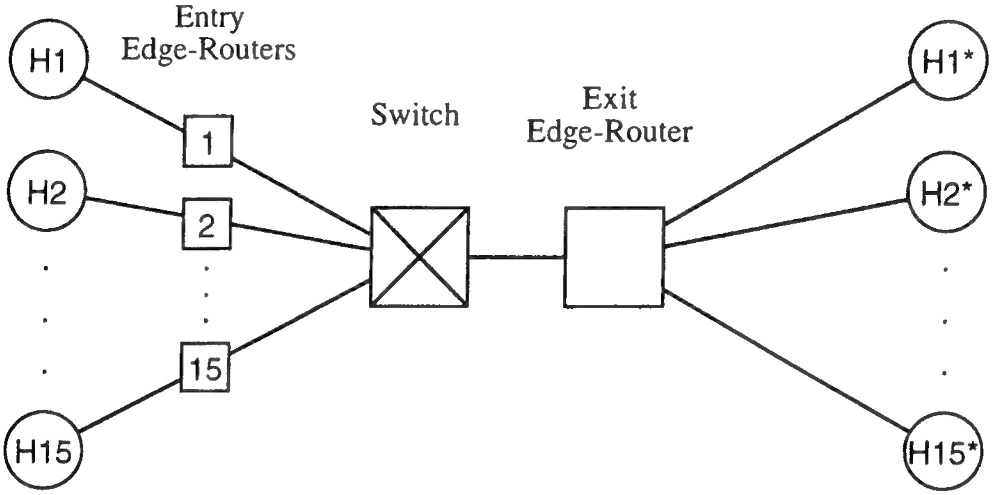

Figure 15: One Switch Network with 15 Sources

All figures show the normalized throughput on the bottleneck link (for the sum of all connections) as a function of switch buffer size for uncontrolled (UBR) traffic and for ABR with different segmentation buffer sizes (each segmentation buffer size is represented by a separate line on the graph). Packet sizes etc. are as described in the introduction to section 7, so that disregarding backward RM cells of the reverse channel the highest possible normalized throughput is 81.6%. Backward RM cells of the reverse channel reduce throughput even further.

It is apparent from Figure 16 (the two source case) that even with the smallest switch buffer of 128 cells, ABR is able to control the flows smoothly enough to prevent congestion at the ATM switch. With a segmentation buffer of 128 cells, the flow from the two sources can only load the ATM link to just under 60%. Throughput increases until the segmentation buffer reaches 3072 cells, at which point the ATM link is saturated.

With more and more active connections the ACRs are varying more and more, and

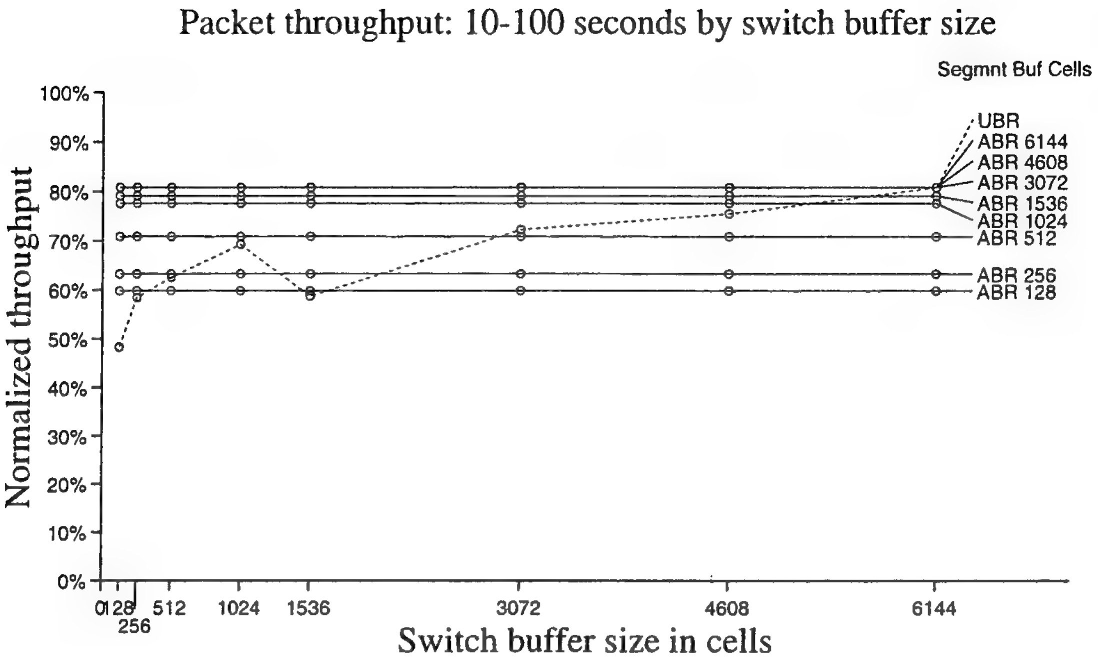

Figure 16: Two Sources – Throughput by Switch Buffer Size

as a result switch buffer overflow occurs more and more often. As a result, the windows of the TCP connections are halved more often and do not grow large enough to make segmentation buffer overflow possible. Hence, the actual size of the segmentation buffer becomes irrelevant for ABR (as long as it is not too small).

There is no segmentation buffer for UBR flow, so the size of the switch buffer is the constraining factor in this case. In general, the flow increases as the switch buffer size increases, reaching the theoretical maximum at a buffer size 6144 cells. There is a notable drop in throughput for a buffer size of 1536 cells. This is apparently due to resonance between the TCP flow control and the details of timing that are affected by the switch buffer size. This may be an artifact of the simulation, which does not introduce random variations in delay that could be expected in an actual system.

Figure 17 shows the effect of adding four additional competing persistent TCP sources.

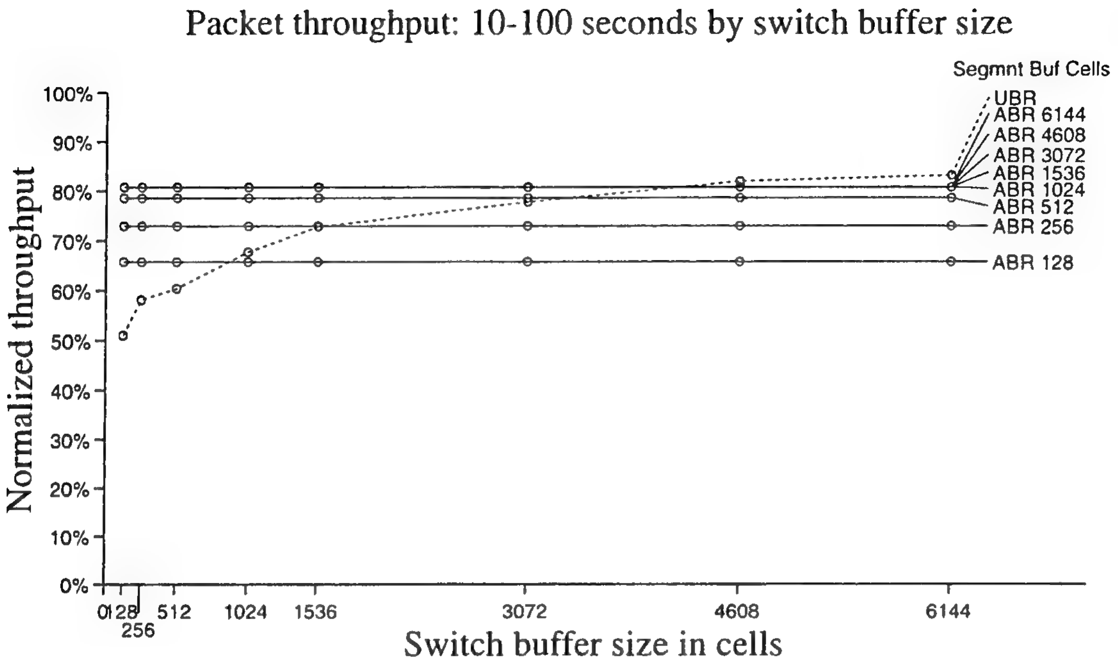

Figure 17: Six Sources – Throughput by Switch Buffer Size

Since each ABR controlled source has its own segmentation buffer, it is not surprising that saturation is reached at approximately one third of the buffer size that was observed for two sources. Once the segmentation buffer reaches 1024 cells, the ATM link is fully utilized (up to the unavoidable overhead). In this case, the UBR throughput increases smoothly with increasing switch buffer size, actually exceeding the throughput of ABR when the switch buffer increases to 4608 cells and above. This phenomenon is due to the RM cell overhead required by ABR. At smaller switch buffer sizes, the smooth flow characteristics of ABR overcome the constrained buffer, but lose their advantage when the switch buffer is large enough to avoid overflow.

The final figure, Figure 18, describes the behavior of 15 persistent TCP sources. As expected, the ATM link becomes fully utilized when the total of 15 segmentation buffers is sufficiently large, in this case about 512 cells per source segmentation buffer. The UBR case again show a strong dependence on the size of the switch buffer, exceeding the ABR throughput at 4608 cells. For this experiment, there is an anomalous point at a switch buffer size of 128 cells that actually increases UBR throughput to a higher level than expected. Again, this appears to be an artifact of the non-randomness in the simulation and resonance effects.

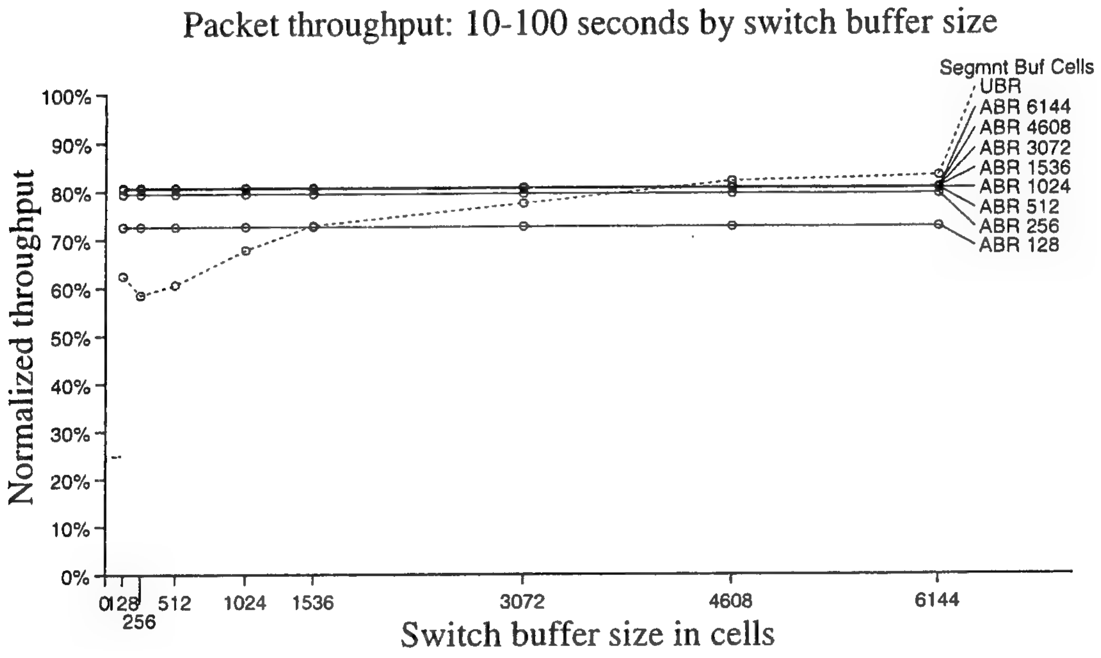

Figure 18: Fifteen Sources – Throughput by Switch Buffer Size

All conclusions must be read with the understanding that they hold only for the architectures simulated. In particular, there is the critical assumption that the TCP/IP hosts do not know about the ACR. If hosts have an ATM card of their own, and integrate TCP/IP and ATM at the host, we would expect very different results. Similarly, changing TCP behavior to TCP-Vegas, or including Selective Acknowledgements in TCP-Reno, could change performance.

We have the following pertinent observations:

(1) Based on the performance curves, no case can be made that for the single –

bottleneck network 15 ABR is superior to UBR in performance (given that switch buffers are growing increasingly large). This observation is would likely be even more obvious if we included features such as "Framedrop" (in the switch), "Drop from Front" and "Random Early Detection", which in our opinion are more likely to favor UBR than ABR. However, we have already seen in section 7.2 that in a multi-bottleneck environment ABR can have considerable advantage in fairness (though not necessarily in total throughput).

(2) The simulation uses a choice of ABR parameters that a mature network might not choose for similar traffic. The way a mature network would be likely to choose ABR parameter values must be studied.

(3) The simulation uses an algorithm for switch behavior that is not likely to be what we will find in mature networks. The problem of finding good algorithms for switch behavior needs attention. Minor variations on the algorithm used in this work may very well be suitable for LANs, but switch behavior for ATM MAN and WAN networks is an area of needed future research, quite possibly connected with the problem of rate setting.

(4) For evaluating the potential for improvements due to improved switch behavior, the most important observation is that in the case of many connections the total throughput becomes virtually independent of the segmentation buffer size. This implies that in the case of many active sources a small improvement in the switch behavior, leading to better convergence of the ACRs of the various VCs, can have a large positive impact on performance.

(5) There may be a need for "signaling" between the packet hosts (source and destination) of a TCP connection and the VC VS, in order not to set the PCR of the VC higher than the port speeds (or modem speeds) of the hosts.

8 Conclusions

The main conclusions we draw from our investigations are:

- **Fluctuating VCs.** If an ABR VC is likely to be fluctuating, i.e. frequently switches from busy to idle and vice–versa, it is desirable the VS has a mechanism that when it goes from busy to idle it sends an RM cell that informs the switches on its path. For this to happen, a change in the ATM Traffic Management Standard seems required. Barring such a change, it is desirable not have fluctuating ABC VCs, or have only a few (presumably high priority, high weight) fluctuating ABR VCs.

- **Fairness of ABR–ER ABR–EFCI and UBR.** ABR ER is quite successful in eliminating the bias against TCP connections with a large Round Trip Time, and the bias against connections traversing many bottlenecks (as long as all those bottlenecks are ABR–ER capable switches). ABR–EFCI is considerably less effective, and UBR strongly suffers from "beat-down" of connections passing through many bottlenecks.

- **Throughput of ABR and UBR.** On a single–bottleneck network we found that ABR has no throughput advantage over UBR. Mostly, it has a small throughput disadvantage.

- **Priorities through weight and through access line speed.** For ABR VCs to profit from a high weight, it is necessary that the VC, or the TCP connection is not already limited by a different constraint, for example its access line speed. If both access line speed and weight are increased, we found that performance always improves, often drastically so. It must be remembered that there may be other factors limiting throughput of a TCP connection, for example high loss which keeps the congestion window small.

- **Locations of delay.** For TCP connections going partially through a packet based network and partially through an ATM (cell based) network, we did not find an effect of where the delay is located.

Interesting follow-up.research will be to investigate algorithms for ABR VSs on when to send an RM informing the switches that the VC is voluntarily reducing its ACR. A related interesting area are algorithms for VSs to "deduce" the actual rate it should ask for if it is lower than the PCR.

Other interesting follow-up research is a combined analysis of fairness and throughput issues between ABR-ER, ABR-EFCI, and UBR.

References

[1] ATM Forum Draft Standard on Traffic Management, Version 4.0, April 1996. ATM Forum/af-tm-0056.000.

[2] Ott, T.J. The Available Bit Rate Service Category in ATM. (Nov 1997). This is a tutorial on ABR written for Rome Laboratories, and delivered to Rome Laboratories in November 1997.

[3] Ott, T.J. Introduction to TCP, in particular TCP Reno. (June 1998). This is a tutorial on TCP written for Rome Laboratories, and delivered to Rome Laboratories in June 1998.

[4] Wong, Larry H. Software User Manual, Item No. A006. (Sept 1998) This is the Manual for the Simulation Tool delivered to Rome Laboratories. Drafts were delivered in July 1997, Dec 1997 and in Jan 1998. The Sept 1998 version will be virtually unchanged.

[5] Mathis, M., Semke, J. Mahdavi, J. and Ott, T.J. (1997) The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *Computer Communications Review 27* (3), pp 67 - 82 (July 1997).

[6] Ott, T.J., Kemperman, J.H.B., and Mathis, M. (1996) The Stationary Behavior of Idealized TCP Congestion Behavior. ftp://ftp.bellcore.com/pub/tjo/TCPwindow.ps

[7] Cunha, C.R., Bestavros, A, and Crovella, M.E. (1995) “Characteristics of WWW Client-based Traces”. Tech. Report BUCS-TR-95-010, Boston University, CS Dept. Boston, MA.

[8] Almeida, V., Bestavros, A., Crovella, M.E. and de Oliveira, A. (1996) "Characterizing Reference Location in the WWW". *Proceedings of of PIDS:* The IEEE Conference on Parallel and Distributed Information Systems, Miami Beach, Florida. December 1996.

[9] Crovella, M.E. and Bestavros, A. (1997) "Self Similarity in World Wide Web Traffic: Evidence and possible causes". *IEEE/ACM Transactions on Networking*, December 1997.

[10] Van Jacobson. Congestion avoidance and control, *Proceedings of ACM SIGCOMM '88*, August 1988.

[11] Stevens, W.R. (1994) *TCP/IP Illustrated*, volume 1, Addison-Wesley, Reading MA, 1994.

[12] Wright, G.R. and Stevens, W.R. (1994) *TCP/IP Illustrated*, volume 2, Addison-Wesley, Reading MA, 1994

[13] Parekh, A.K. and Gallagher, R.G. (1993) " A generalized processor sharing approach to flow control in integrated services networks: The single node case", *ACM/IEEE Transactions on Networking*, Vol 1, No. 3, June 1993.

[14] Golestani, J. (1995) "Network Delay Analysis of a Class of Fair Queueing Algorithms", *IEEE Journal on Selected Areas in Communications*, Vol 13, No. 6, Aug 1995.

[15] D Stiliadis, D. and Varma, A. (1998) "Rate-proportional servers: A design methodology for fair queueing algorithms", *ACM/IEEE Transactions on Networking*, Vol 6, No. 2, April 1998.

[16] Kevin Fall and Sally Floyd, Simulations-based comparisons of tahoe, reno and SACK TCP, *Proceedings of ACM SIGCOMM '96*, May 1996.

[17] Janey C. Hoe, Improving the start-up behavior of a congestion control scheme for TCP, *Proceedings of ACM SIGCOMM '96*, August 1996.

[18] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson, TCP vegas: New techniques for congestion detection and avoidance, *Proceedings of ACM SIGCOMM '94*, August 1994.

[19] Lawrence S. Brakmo and Larry L. Peterson, Performance problems in BSD4.4 TCP, *Proceedings of ACM SIGCOMM '95*, October 1995.

[20] Sally Floyd, Connections with Multiple Congested Gateways in Packet-Switched Networks Part I: One Way Traffic. *CCR* **21** no 5 pp 30 - 47.

[21] Sally Floyd, TCP and successive fast retransmits, February 1995, Obtain via ftp://ftp.ee.lbl.gov/papers/fastretrans.ps.

[22] Sally Floyd and Van Jacobson, Random early detection gateways for congestion avoidance, *IEEE/ACM Transactions on Networking*, August 1993.

[23] Sally Floyd, TCP and explicit congestion notification, *ACM CCR*, 24(5), October 1994.

[24] Matthew Mathis, Jamshid Mahdavi, Sally Floyd, and Allyn Romanow, TCP selective acknowledgement options, May 1996, Internet Draft ("work in progress") draft-ietf-tcplw-sack-02.txt.

[25] Matthew Mathis, Jamshid Mahdavi, Forward Acknowledgment: Refining TCP Congestion Control, *Proceedings of ACM SIGCOMM '96*, August 1996.

[26] Lakshman, T.V., and Madhow, U. (1997) The Performance of TCP/IP for Networks with high Bandwidth-Delay products and random loss. *Trans of Netw* 1997.

[27] Lakshman, T.V., Madhow, U. and Suter, B. (1997) Window-based error recovery and flow control with a slow acknowledgement channel: a study of TCP/IP performance *Infocom '97.*

☆U.S. GOVERNMENT PRINTING OFFICE: 2000-610-055-10037

MISSION OF AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of Information Systems Science and Technology to meet Air Force unique requirements for Information Dominance and its transition to aerospace systems to meet Air Force needs.